06

Los primeros pasos con JDBC

Transcripción

[00:00] Hola. Ahora sí vamos a la acción. Ya entendimos cómo una aplicación Java se comunica con la base de datos y podemos poner en práctica lo que aprendimos hasta el momento. Aquí en el Eclipse vamos a crear un proyecto Maven, entonces vamos a hacer un file new project. Aquí voy a escribir el Maven, Maven Project. Create simple project.

[00:20] Hago un next acá, el group va a ser com.alura. El artifact va a ser control-de-stock y hago un finish. Listo, ya creamos el proyecto. Ahora vamos a hacer un clic derecho aquí en el proyecto, y vamos a properties. Y aquí, cuando abrimos las propiedades del proyecto, vamos aquí a la opción de Java build path. ¿Por qué?

[00:50] En mi caso, siempre que estoy creando un proyecto en eclipse, me está creando con la versión 1.5 de Java y aquí lo vamos a hacer un cambio para que sigamos con la versión 11 como habíamos combinado. Entonces elijo aquí la opción del Java 1.5, hago un clic en edit y elijo acá el workspace default. Ya tenemos aquí con la versión 11. Hago un finish, apply and close y listo.

[01:18] Ahora sí tenemos el proyecto seteado para la versión 11 de Java. Ahora lo que vamos a hacer es expandir acá el proyecto y vamos a abrir el archivo pom.xml. Aquí en este archivo tenemos todas las configuraciones del proyecto, y como había comentado cuando nosotros estamos agregando dependencias del proyecto, por ejemplo la librería de conexión del driver de conexión con MySQL, nosotros tenemos dos formas.

[01:45] O podemos utilizar el archivo .jar que bajamos desde internet para agregar aquí manualmente el proyecto o nosotros podemos utilizar herramientas de control de dependencias, de manejo de dependencias, como es el caso de Maven. Aquí vamos a hacer unas configuraciones iniciales que van a ser las siguientes.

[02:05] Aquí ya tenemos algunas y nosotros vamos a agregar unas cositas más, que va a ser el name del proyecto que es control de stock, una descripción que va a ser proyecto para trabajar con bases de datos con JDBC. Vamos a poner aquí también, vamos a declarar unas profiles no, perdón, vamos a declarar las propiedades.

[02:36] Y aquí, en realidad, no son propiedades, pero es una propiedad adentro del listado de propiedades. La propiedad va a ser Java.version con el valor 11 de la versión 11 de Java para hacer el build de la aplicación. Ahora sí vamos a declarar aquí una tag de dependencias que por ahora, va a estar vacía, pero va a ser aquí en donde vamos a estar agregando todas las librerías que vamos a estar utilizando en este proyecto.

[03:06] Ahí está, las dependencias. Y por último, vamos a estar configurando la forma de hacer el build de la aplicación adentro de la tag de build. Aquí vamos a agregar un listado de plugins con esta tag. Y adentro de ella vamos a agregar un plugin para hacer la compilación del proyecto.

[03:27] Este plugin va a tener las siguientes propiedades, un groupId que se llama org.apache.maven.plugins. El artifactId va a ser maven-compiler-plugin. Tiene una versión también. Entonces agregamos aquí la versión 3.7.0. Y por último, agregamos unas configuraciones que son las siguientes. Va a ser el source que vamos a agregar aquí la variable java.version.

[04:10] La próxima va a ser el target, que también vamos a copiar este valor. Entonces voy a copiar acá de arriba y lo pegó aquí abajo. Y por último, una propiedad de optimización que va a estar prendida con el valor true. Ahora

aquí en el proyecto, vamos a crear un paquete. Va a hacer un clic acá, "Ctrl + N" package. Vamos a crear un paquete llamado com.alura.tests en donde vamos a estar creando nuestras clases de pruebas para hacer las operaciones con la base de datos.

[04:46] Y ahí vamos avanzando con el proyecto. Aquí, en este paquete, vamos a estar creando una clase. Esta clase va a llamarse pruebaConexion porque es lo primero que vamos a estar haciendo, vamos a probar la conexión como habíamos visto en el dibujo de la clase anterior. Aquí tenemos la prueba conexión y vamos a crear un método llamado main, en donde vamos a estar poniendo nuestra lógica.

[05:11] Hasta el momento, nada nuevo. Vimos aquí creamos un proyecto Java, creamos una clase, un método main o más nuevo puede ser esto del pom.xml, para manejar las dependencias del proyecto, agregar librerías, todo tranquilo hasta ahora. Ahora vamos a traducir los dibujos que vimos en la clase anterior para el código.

[05:35] Entonces, vamos a estar intentando conectarnos a la base de datos que creamos anteriormente, utilizando el concepto que aprendimos en la clase anterior. Nosotros aquí necesitamos crear una conexión con un objeto de la interfaz connection que habíamos visto ahí en la clase anterior. Y ahora vamos a estar haciendo la conexión con la base de datos, como aprendimos en la clase anterior.

[05:57] ¿Y qué necesitamos aquí? Nosotros necesitamos de una variable para recibir esta conexión y esta variable es justamente del tipo connection, del paquete Java MySQL, voy a decir aquí connection con y ¿cómo tomamos esta conexión con MySQL? ¿Recuerdan la clase anterior? Nosotros utilizamos la interfaz DriverManager.

[06:28] Esa es la clase que vamos a estar utilizando para tomar la conexión con la base de datos. ¿Y con cuál método? El getConnection. Y aquí vimos que

tenemos tres opciones para conectar a la base de datos con el getConnection. Nosotros vamos a estar utilizando esta de acá, la que tiene tres parámetros, la de URL, usuario y la contraseña.

[06:53] Aquí, en el primer parámetro vamos a estar agregando la URL de conexión con la base de datos, quiero hacer la siguiente: "Jdbc:" ¿se acuerdan del formato que vimos en la clase anterior? Mysql, que es la base de datos que vamos a estar utilizando, dos puntos, barra, barra, localhost, acá no ponemos el puerto porque no es necesario, control_de_stock.

[07:20] Y aquí vamos a hacer, vamos a agregar unos parámetros que vimos en la clase anterior que eran opcionales, pero aquí vamos a agregar un parámetro para dejar configurado, bien configurado aquí el time zone. Entonces vamos a decir aquí la interrogación para parámetros, "useTimeZone=true&serveTimeZone=UTC". Ahí está.

[07:52] Para el usuario vamos a poner "root", que fue el que configuramos cuando instalamos la base de datos. Y para la contraseña va a ser "root1234" que es la contraseña que también configuramos cuando instalamos el MySQL. Bueno, escribimos aquí el comando para tomar la conexión y Eclipse ya puso todo rojo, ya se queja de algo que escribimos acá, ¿qué será que hicimos mal?

[08:20] Bueno, acá dice que tenemos una excepción que no fue tratada, la SQL exception. Eso es porque cuando nosotros vamos a conectarnos a una base de datos pueden ocurrir errores, muchos tipos de errores y excepciones. Por ejemplo, la base de datos puede estar offline, nosotros podemos estar intentando conectar a una URL inválida, puede estar faltando algún componente, podemos estar equivocados como con las credenciales de login del usuario.

[08:45] Entonces en cualquier situación de error será lanzada una excepción. Aquí podemos hacer dos cosas, o nosotros tratamos la SQL exception con un try catch, como nos dice acá, o nosotros podemos lanzar la excepción con un

throws. Aquí nosotros vamos a hacer un throws ahora porque no tiene mucho sentido que estemos tratando ahora esta excepción.

[09:12] Entonces vamos a agregar la declaración de throws exception acá de SQL Exception. Ahí está. Eclipse paró de quejarse. ¿Y ahora qué vamos a hacer con la conexión? Recién la tomamos y ahora la vamos a cerrar con el comando close.

[09:29] Y eso es muy importante, porque siempre que se ha abierto una conexión o cualquier otro tipo de recurso, por ejemplo un archivo, los mismos tiene que ser cerrados después de utilizados, para que no queden ahí abiertos y puedan corromper alguna cosa. ¿Y ahora qué vamos a hacer con esta conexión? Nosotros la vamos a cerrar, vamos a hacer así aquí un close.

[09:52] Y ya está, con eso hacemos la prueba de conexión. Y es super importante que siempre que abramos una conexión, nosotros enseguida, después de utilizarla la cerremos. Primero para liberar el recurso y segundo, para no estar comprometiendo la base de datos. Lo mismo podemos hacer con otros tipos de recursos, por ejemplo un archivo.

[10:14] Si nosotros no lo cerramos después de utilizarlo, otras aplicaciones y otros procesos no lo pueden utilizar, acá es igual. Entonces, no se olviden, abrieron una conexión, cerramos luego de utilizarla. Listo. Acá finalizamos el código, vamos a guardarlo, Eclipse y ya hizo el build y ahora lo vamos a ejecutar, entonces hagamos aquí un clic derecho, run as Java application.

[10:43] Y en la consola ya tenemos una exception. Mal pusimos el throws, ya tenemos nuestro primer error. Vamos a ver qué dice acá en la consola. Bueno, acá nos dice que no es un driver para conectarse con la base de datos. Se acuerdan que vimos en la clase anterior que habíamos visto que para que nuestra aplicación Java pueda conectarse a una base de datos de MySQL, nosotros tenemos que agregar el driver de MySQL al proyecto y eso nosotros no lo hicimos todavía.

[11:19] Entonces vamos por ello. Vuelvo acá, vamos al pom.xml perdón y aquí en la tag de dependencies vamos a agregar una nueva dependencia con la tag dependency. ¿Qué va a tener él? GroupId de MySQL, y el artifactId va a ser mysql-connector-java. Y la versión va a ser la misma de nuestra base de datos instalada, la 8.0.26. Ahí guardé el archivo. Ahora vamos a hacer la prueba una vez más.

[12:00] Vamos a ver acá, venimos aquí al proyecto, a la clase, hacemos un ran as otra vez. Y ahora la consola no presenta nada, no hay ningún error, o sea, parece que está todo bien. Vamos a hacer lo siguiente. Vamos a agregar aquí un system out, syso, que es un acceso directo para el System.out.println y vamos a decir que estamos cerrando la conexión.

[12:34] Ahora sí, vamos a ver si el comando realmente fue ejecutado, si no pasó nada. Vamos a ejecutar una vez más y tarda un poquito. Ahí está, cerrando la conexión, o sea, funciona todo bien. Ya empezamos a dar los primeros pasos en el proyecto y aprendimos en la práctica cómo abrir y cerrar una conexión utilizando las interfaces de JDBC.

[13:00] Con esta lib dejamos nuestra implementación transparente, independiente de la base de datos elegida, por ejemplo, podría ahora estar agregando el driver de SQL Server y me conectaría igual, cambiando la URL de conexión. Ahora que tenemos la conexión, vamos a poder realizar las operaciones de SQL de insert, update, delete y select. Y eso vamos a ver en la próxima clase.

[13:26] Entonces en la próxima clase nosotros vamos a estar tomando como base una aplicación sencilla para escribir las funcionalidades de ella que van a hacer la gestión de la conexión y la manipulación de su contenido.